Как перестать волноваться и начать ревьюить код быстро и эффективно

"эффективно" - имеется в виду так чтобы была польза пользователю, автору, ревьюеру и мейнтенерам

Почему я об этом решил рассказать?

- а я-то собственно кто такой?
  - Антон Дудаков
    - разработчик
      - в Zina
        - это такой финтех стартап в Дубае
          - с командой из таких компаний как
            - Яндекс
              - вот тут я работал
                - и Вова тоже
            - СберДевайсы
              - и тут я работал
            - Apple
              - а тут работал наш СТО
                - программистом кстати, это вдруг вы подумали что просто так хочу прерываюсь
            - Uber
              - а тут бекенд лид
        - а ещё
          - мы ищем бекендера
      - в основном для мобильных
      - в основном Android
    -

    - иногда выступаю на митапах и конференциях
      - если вы в Дубае, то приходите, будем вместе делать митапы для мобильных разработчиков
- менять мир к лучшему
  - Вы тратите меньше энергии
    - вы счастливы
  - Вы сохраняете хорошие отношения
    - ваши коллеги счастливы
  - код в чистоте
    - ваши последователи счастливы
  - высокий уровень качества продукта
    -

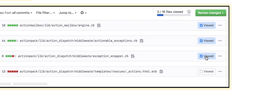
    - пользователи счастливы

ответственность

- что такое ответственность
  - нужно разделять причину и ответственность
    -
      - причина это про то чьи изменения поломали нетривиальную работу
    -
      - ответственность это про тому кому чинить
- на ком она?
  - на обоих
    - чинить - если не первому то второму
    - не забывайте об этом во время ревью
      - а ещё не забывайте о вашем пользователе, ему вообще этим пользоваться

структура и последовательность

- разные подходы
  - review-tool based
    - последовательное ревью всех файлов
      - плюсы
        - над последовательностью не нужно думать
        - вы точно ничего не пропустите
      - минусы
        - сложнее понять что именно делает код
          - а значит и находить ошибки если они размазаны по разным блокам
  - feature-based
    - подход
      - зависимые блоки
        - находим точку входа в фичу и начинаем разматывать клубок
          - у мобильников это обычно экран
            - или кнопка перехода на экран
              - да ещё и файл навигации
        - тесты могут помочь понять что именно происходит
        - скачать себе ветку
          - чтобы понять архитектуру и зависимости
      - независимые блоки
        - это разные файлы, которые не используются нигдеже в логике
          - extensions
            - удобно когда они в отдельных файлах
          - utils
            - форматтеры
            - валидаторы
            - запускаторы
          - resources
            - ага, их тоже будет не лишним пробежать глазами
              - дубликаты
              - не экранированные символы
              - хардкоды, вместо темплейтов
                - или split + concatenate
                - `let mainScreenHelloMessagePart1 = "Hello, "`
                - `let mainScreenHelloMessagePart2 = "!"`
                - `let mainScreenHelloMessage = "Hello, %1$@!"`
              - нарушение соглашения об именовании ключей
                - у вас же есть такое?
                - ну хотя бы неофициально?
              - растр, там где можно вектор, и вектор там где лучше растр
                - часть из этого линтится
    - плюсы
      - глубже погружение в фичу
        - структура кода
          - в том числе зависимости
        - необработанные случаи
    - минусы
      - сложнее найти с чего начать
        -

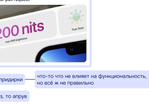
      - есть риск пропустить что-то, что не связано с фичей
        - кажется это решается галочками
          - 

ВНИМАНИЕ

- на что тратить
  - в меня точно добавили не по ошибке?
    - можно добавить владельца данного кода
      - кое где даже есть возможность добавлять владельца автоматически
  - делает ли код то что заявлено?
    - (скриншот/видео)
      - было бы неплохо сравнить с макетом
    - а все ли edge cases учтены?
  - архитектура
    - понятно ли как взаимодействуют разные части системы?
    - интегрируется ли в остальную систему
    - расширяем/поддерживаем ли код?
      - а должен ли он таким быть
  - следите за соблюдением тех правил, которые есть у вас в компании и в ваших сердцах
    - сравните с вашим код-стайлом
      - кстати, а почему вы ещё не автоматизировали эти проверки?
    - или на то как принято делать в языке
      - избегайте nullable-типов и !!
        - обрабатывайте nullable по хитрому
          - swift
            - `if let nonNullableValue = optionalValue { // ... }`
            - `if let optionalValue = optionalValue { optionalValue.call() }`
            - `guard let nonNullableValue = optionalValue else { return } // ...`
            - `if caseNumber == .caseValue(let smth) { smth.doMe() }`
          - kotlin
            - `optionalValue?.let { nonNullableValue -> // ... }`
            - `val nonNullableValue = optionalValue ?? return // ...`
      - предпочитайте val (let) вместо var
      - используйте when вместо многосложных if-ов
        - ну и используйте if-ы когда вам нужно проверить только одно условие
        - Но для обработку enum-ов всегда используйте when
      - и так далее
    - можно ориентироваться на разные умные книжки, например на Clean Code
      - убирайте неиспользуемый код
        - лучше конечно это автоматизировать
      - не используйте комменты в коде
        - лучше делайте код понятным
          - "говорящие" имена методов и переменных
          - принцип единой ответственности
      - visibility
        - по дефолту всё private
      - использование значений вместо именованных констант
      - ну и так далее
        - подробнее смотри в книге: Clean Code by Robert Martin
    - тесты
      - ваши политики соблюдаются?
        - они же у вас есть, да?
        - и покрытие кода измеряете, правда?
      - а тесты точно работают?
      - а все кейсы покрывают?
  - так называемый common sense
    - дублекация кода
    - захардкоженые значения
  - concurrency
    - гонки?
    -
      - дедлоки?
  - сложность
    - не переусложнён ли код
    - не занимается ли автор изобретением "велосипеда"
    - не занимается ли решением задачи, которая ещё не появилась?
      - в рамках данной задачи
      - может эту задачу прямо сейчас уже кто-то другой решает
  - документация
    - иногда надо
      - ну например если это публичный API
  - обработка ошибок
  - наличие логов
    - и аналитики
    - и полезной информации в них
      - что случилось
      - в каком месте приложения
      - какое было окружение в этот момент
    - и отсутствие в них персональных данных пользователей
- а на что нет
  - код-стайл
    - идите уже и автоматизируйте эти проверки, ну ёмаё
  - отформатированный код
    - он точно нужен в репозитории?
- спорно
  - переименования
    - нужно отделить переименование от всего остального
      - было бы круто если бы автор сделал это отдельным коммитом
        - или даже отдельным пулреквестом
- а что ваще оставить автору
  - его авторские решения
    - если они не противоречат тому как принято на проекте

feedback и взаимоотношения

- не забывайте хвалить автора за интересные решения
  - да и даже и за обычные куски которые принесли кусочек тепла в ваше сердце
    - 🤗
- если нашли ошибку
  - опишите как она проявится
  - предложите свой вариант кода
    - вы же понимаете, да?
      - вы оба ответственны за PR
    - в гитхабе для этого есть удобный механизм
      -

- если не согласны
  - объясните почему, ссылитесь на что-то
    - книжка
    - статья
    - ваш документ по код-стайлу
    - часть кода из вашего же проекта
    - желательно прямо ссылку приложить
- используйте пометку "nits: "
  - 

  - nitpicking - придирки
    - что-то что не влияет на функциональность, но всё ж не правильно
- апрув с ошибками
  - если только nits, то апрув
  - если не только nits, то тоже можно
    - но если вы автору полностью доверяете
      - это мало ли он проносит кота в машину
- финальный коммент
  - не держите в себе положительные эмоции
    - выскажите автору как вы рады что данная фича наконец-то сделана
  - если эмоции по поводу качества кода переполняют, то помогите автору стать лучше :)
    - ссылки на статьи, книжки, гайды компании
    - но то есть всё вышеописанное

последний "слайд"

- что почитать
  - "Clean Code" by Robert Martin
  - Effective $yourFavLang
  - Google's code reviewers guide
    - https://google.github.io/eng-practices/review/reviewer/
- контакты
  - /hwdude
    - twitter.com
    - t.me

Presented with xmind